International Journal of Computer Science Engineering
and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 4, Issue 2, Apr 2014, 51-58

ADVANCED APPROACH IN QUERY REFINEMENT USING REFINEMENT FILTERS FROM KNOWLEDGE BASE

RALLA SURESH[1] & BIKASH CHANDRA ROUT[2]

[1]Research Scholar, Department of CSE, Osmania University, Hyderabad, Andhra Pradesh, India

[2]Rourkela Institute of Management and Technology, Rourkela, Odisha, India

ABSTRACT

Web search engines today often complement the search results with a list of related search queries. Query refinements, particular kind of related queries, are obtained by finding queries that are most likely to follow the original query in a user session. A refined query is a query that is created based on a refinement option selected by the user, for example, to limit the query to a specific data range. This algorithm clusters refinements based on their likely underlying user intents by combining document click and session co occurrence information. To improve the selection and placement of the query suggestions proposed by a search engine, and can also serve to summarize the different aspects of information relevant to the original user query.

KEYWORDS: Query Refinement, Sessions, Search Engine, Refined, Query

INTRODUCTION

The search query is a set of words or phrases a user enters when looking for information on a specific topic or subject. Formulating a search query is a challenging task for most of users because they are required to express their anomalous states of knowledge. Users frequently modify a previous search query in hope of retrieving better results.

Therefore, it may not be easy for users to choose the query terms that represent precisely the topic they are looking for. As a result, users sometimes need to modify their queries based on the search results retrieved. we differentiates these two stages as the initial query formulation stage for constructing search strategy, and the query reformulation stage for adjusting the initial query manually or with the assistance of the Knowledge base. Web users show very different patterns of searching from those found in the traditional information retrieval systems such as online databases. For example, most Web users do not have many queries per search, and each query tends to be short. Many web search engines today offer query reformulation suggestions by, for example, mining query logs of knowledge base. Users are manually reformulating their queries based on the search results from the initial query, and their knowledge and experience of how search engines work. The reformulation process is an iterative endeavor between users and search engines in getting a satisfactory set of results. A query result typically presents refinement options in the form of string values or value ranges. Each string value or numeric value range is known as a refinement bin. For each refinement bin, there is an associated Refinement Token value. The Refinement Filter element represents one refinement bin. The resulting query is limited to items where the associated managed property has a value within the refinement bin.

We can add multiple refinement bins in the Refinement Filters element. In this case, the resulting query will be limited to items where the associated managed property has a value within all refinement bins. This enables you to apply refinement on multi-valued properties. For instance, refine the query to items that have two authors (each represented by a refinement bin), but exclude items that have only one of the authors. One refinement bin represents a specific value or value range for the managed property. The goal of this work is to look at the types of query reformulation users perform and evaluate them using effectiveness metrics such as click data. In order to study these metrics, we first construct taxonomy of query reformulation strategies adopted by users. Next, we build a Refinement filter for these different types of reformulations. While there are some existing classifiers that determine whether a query is a reformulation, ours is the first to separate them into reformulation types.

Specification: Users specify the meaning of the query by adding more terms or replacing terms with those that have more specific meaning.

Generalization: Users generalize the meaning of the query by deleting terms or replacing terms with those that have more general meaning.

Replacement with Synonym: Users replace current terms with words that share similar meaning.

Parallel Movement: Users do not narrow or broaden previous queries. The previous queries and the follow-up queries have partial overlap in meaning, or these two queries are dealing with somewhat different aspects of one concept

RELATED WORK

Much of the work on query reformulation for web search has focused on offering automatically generated query suggestions to the user. The suggestions are typically shown on the same page as the search results. These query suggestions are built into every major search engine today. Prior research in this vein has explored computer-generated suggestions using query expansion, query substitution, and other refinement techniques .Implicit relevance feedback from users is a common data source for computer-generated reformulations. For example, work by Baeza-Yates *et al.* [2] uses query logs to discover new query refinements, finding similar queries using a cosine function over a term-weighted vector built from the clicked documents. A study showed that these automatically generated refinements were as effective as human constructed reformulations, using metrics such as uptake and click behavior.

Click Data Analysis

Many researchers have studied click data as indicators of search relevance. An early inquiry by Joachims [19] reveals that click data can indeed be used to improve search relevance. Several later studies agree that click data are indicators of search result preferences and discuss best methods of analyzing click data [2][12]. Joachims *et al.* also find that analyzing clicks over query refinements similarly provides useful information [11]. This data has also been shown to be helpful for improving search relevance [1][9]. Our study applies lessons learned from these reports of click data analysis. While they study the effectiveness of analyzing different click Patterns, we study the effectiveness of refinement strategies using different click patterns.

Taxonomies of Refinement Strategies

Taxonomies of query refinements have been developed for different types of search. A more comprehensive review of query in traditional information retrieval can be found in [10]. Here we focus only on the taxonomies developed by analyzing query logs. These are generally constructed by examining a small set of query logs. Some studies are out of date or incomplete. None have built automatic classifier distinguishing refinement strategies, as. Anick [1] classified a random sample of 100 refinements by hand into eleven categories. Lau and Horvitz [8], Jansen *et al.* [5], and He *et al.* [4] used the same refinements categories—terms taken from linguistics [7]. As part of a study of re-finding behavior, Teevan *et al.* constructed a taxonomy by looking through query logs, and implemented algorithms to detect a subset of the refinement strategies. Whittle *et al.* modeled some refinement strategies using a graphical network. Bruza and Dennis [3] manually classified 1,040 queries into their own taxonomy. Guo *et al.* [6] also constructed a small taxonomy and used a conditional random field model to predict query refinements. Riehand Xie [13] constructed conceptual reformulation categories like content, format, resource; these are not included in the table because their abstract nature makes them difficult to map against concrete refinement techniques.

Refinement Strategies

We constructed our own taxonomy by combining the types of query refinements identified in prior work. We implemented a matching rule for each strategy, which was iteratively improved to find the best unsupervised algorithm. For instance, the 'add words' rule was modified to detect added words even when the other words were reordered. To determine if we were missing any other rules or needed to adjust existing rules, we ran our classifier over the AOL query logs and randomly checked the output. We optimized for reducing false positives while keeping false negatives low since we wanted a high precision classifier. From this, we tweaked several rules and added one that would detect a number of query reformulations that other rules did not, namely *substring*2. A few categories from prior work were either vague or difficult to detect, marked *not Detected* in the table. For example, determining whether a query was a *location* reformulation as defined in [3] is subjective and would reduce the precision of our classifier. Categories marked *not in data* could not be classified because the queries were normalized (via lowercasing and punctuation removal) in our dataset.

MODEL AND DEFINITION

Knowledge Base for Refinement Filters

A Knowledge Base is a structure KB: = (CKB, RKB, I, lc, lr) consisting of:

Two disjoint sets CKB and RKB

- A set I whose elements are called instance identifiers (or instances or objects shortly)
- A function lC: CKB → I called concept instantiation
- A function lR: RKB → I + called relation instantiation

A relation instance can be depicted as r(I1,I2 ,...,In), where r→RKB, Ii →I. Similarly (Ii), where c→CKB, represents the concept the instance Ii belongs to r is called a predicate and Ii is called a term. Note that in this work we treat only binary relations. However the extension for n-ary relations is straight forward.

Query

A conjunctive query is of the form or can be rewritten into the form: Q(X) ≡ for all X P(X, k), with X being a vector of variables (X1,..., X n), k being a vector of constants (concept instances), P being a vector of conjoined predicates (relations). For example, for the query "for all x, y works in(x, KM) and research in(x, y)"

we have X: = (x, y), k := (KM), P:= (P1, P2), P1(a, b, c):= works In (a, b), P2(a, b, c):=research In(a, c).

Since a predicate constrains the interpretation of a variable in a query, in the rest of the text we will use the term query constraint as the description of a predicate. For example, research In (x, y) is a constraint for the interpretation of the variable x.

Moreover, our limiting focus to conjunctive queries is not a serious limitation since the result of a disjunctive query can be considered as the union of the results of the disjoints; that is, each disjoint can be considered as an independent query.

Answers (Results) of a Query

Let ∑ be the set of all relation instances which can be proven in the given knowledge base.

For a query Q= "for all X P(X, k)" an answer is an element (tuple) in the set R(Q) = {X} ={(x1,x2 ,...xn)}, such that P(X, k) is provable, i.e. each of the relation instances r(x1, x2,...,xn, k1,...kl), r →Pexists in the set ∑. If a query cannot be proven (i.e. it returns zero results) it is called a failing query.

Extended Query Refinement Operator

An extended query is a pair <Q, A>, where A is a Boolean function, called Acceptance Test, that takes as input the answer generated by executing the query Q over a knowledge base KB and returns {True, False}. We say that an extended query is acceptable for the KB if A(R (Q)) = true. The concept acceptance Test is introduced as a possibility to express constraints in user's queries that cannot be expressed in the relational language, for example that the user is willing to accept answers from a modified query.

The goal of a query refinement operator is to transform a query <Q, A> to a query<Q', A> so that latter is acceptable. An extended query <Q', A> refines <Q, A> (in the notation <Q, A>→ ref <Q' A>, or as a shorthand (Q →ref Q'), if in the context of the given knowledge base KB and the ontology O holds R(Q') ⊆ R(Q). It is clear that a refinement operator (→ref) derives a set of refinements for a query Q, or more formally:

Q→ref {Q'| R (Q') R (Q)}.

Logic-Based Query Refinement Operator

From the model-theoretic point of view, the query refinement process can be treated as a logic implication, i.e. a query Q' implies another query Q (Q logically entails Q'). Since the ontology and the schema are the only constraints used for driving the refinement process, then (Q →ref Q') ≡ (KB, O ├ Q' →Q), Where ├ depicts the derivation (inference) process.

However, due to the complexity of subsumption reasoning, we use an alternative, more tractable generality order, θ-subsumption, frequently used for efficient implementation of inductive logic programming tasks.

The logic-based refinement of a query Q, denoted $\rho_l(Q)$ is a query obtained in one of the following ways:

- $\rho_l(Q) = Q\theta$, where θ is a substitution that corresponds to the variable x from Q;
- $\rho_l(Q) = (Q \wedge L)\theta^{-1}$ some θ^{-1} (including $\theta=\{\}$), where θ^{-1} is some inverse Substitution and L is a ground literal which is true in KB and there is a term t such that $t \rightarrow Q$ and $t \rightarrow L$

In both cases, (a) and (b):

- Equ refinements, $R(\rho_{\text{logic-based}}(Q)) = R(Q)$, are aggregated to the initial query (i.e. they are treated as equivalent queries to the initial query) and
- Non-minimal refinements, refinements which are subsumed by another (but not equ) refinement A, $R(\rho l_{\text{ogic-based}}(Q)) \subset R(A)$, are filtered.

Query Refinement

The precondition for applying the logic-based query refinement operator ρl is the existence of an ontology-based query. It means that our approach can be directly applied on the queries in the form similar to (1). However, in this case study we can benefit from the combination of the free-text search and the logic-based search since the former is characterized by the very high recall (but low precision), whereas the second is characterised by the very high precision. Therefore, if we apply the logic-based refinement not on the ontology-based query generated from a keyword-based query, but rather on the results retrieved by the free-text search engine, we get the opportunity to generate the candidates for the refinement from a larger set of instances. Consequently, we get more precise refinements. Moreover, since we avoid performing an ontology-based query on the whole knowledge base, we reduce the time needed for generating refinements.

From the conceptual point of view this procedure is possible, since our query refinement has a model-theoretic interpretation, i.e. it can be interpreted through the set of a query's results. From the technical point of view the approach is feasible, since the identifiers of the publications are shared between the database and the knowledge base.

THE EVALUATION

In order to prove the usability and validity of our approach we performed several evaluation studies. proving the usefulness of the query refinement service. This experiment is a classical user-driven study, in which we wanted to prove the user's acceptability of the proposed approach. We compared the effectiveness of the searching with and without the query refinement approach. Since it was difficult to define in advance a set of relevant resources for a query regarding the given repository, we set-up an experiment in which each participant had to perform an unsupervised searching process. More precisely, each participant had to choose ten tasks on his own and a half of them to perform using the standard query interface and another half using the query refinement support. For each task a participant should find five relevant results 5. In order to take into account the quality of selected relevant results, each participant had to express his confidence in these results. The confidence describes a participant's sureness that the selected results are the best possible ones (i.e. that there is no a better result for his need). It is measured on the scale 1 - 4, whereas 4 means maximal confidence. Beside

confidence, we measured the length of a query session (number of querying steps) and the duration of a query session for each query. We selected the 15 participants, who have been undergraduate students in Computer Science.

Each of them made query related to the research he is familiar with. No additional instructions were given to them.The Results of the First Experiment are presented in Table 1

Table1

Method for Querying	Duration of Session(in sec)	Length of Session (Num of Steps)	Confidence
Query Refinement	36282	568	323
Standard	5760	337	226

DISCUSSIONS

Although searching supported by our approach required more querying steps for a task (column 3) it is performed faster (column 2). Moreover, participants were more confident in results found by using the query refinement support in querying (column 4). This means that our approach provides refinements that are very useful (relevant) for a current query, since a user did not spend much time in a querying step. Finally, the knowledge-based approach covers a large part of the searching space with such refinements, so that a user is very confident with results selected as relevant, i.e. user feeling that lots of alternatives are taken into account in the querying process. This is a very important feature in recommender applications a user should trust the recommendation process. In order to prove if these differences can be considered statistically significant we performed a paired t-test for each measure. It did reveal the superiority of our approach with respect to all three parameters ($p < 0.0001$).

RELATED WORK

Using lattices for query refinement process is not new, as some lattice representations were used in for refining queries containing Boolean operators. However as these approaches typically rely on a Boolean lattice formalization of the query, the number of proposed refinements may grow too large even for a very limited number of terms and they may easily become semantically meaningless to the user. These limitations can be overcome by using concept lattices.

We described an approach, named REFINER FILTER, to combine Boolean information retrieval and the content-based navigation with concept lattices. For a Boolean query REFINER FILTER builds and displays a portion of the concept lattice associated with the documents being searched centered around the user's query. The cluster network displayed by the system shows the result of the query along with a set of minimal query refinements/ enlargements.

A similar approach is proposed in, by adding the tance between queries in the lattice is used for similarity ranking. Conceptually, the most similar approach to our query refinement system is the Query, an approach for the navigation through a hyper index of query terms. The hyper index search engine aims users to add, delete or substitute a term from the initial query by providing the minimal query refinements/enlargements. It is designed specifically to (i) help user to express a precise description of their information need and (ii) reduce information overload by presenting the search result at a higher level of abstraction. Moreover, in the analogy between the lithoid, a crystalline structure which organizes document descriptions (and may be used to support searchers in formulating their information demands via Query by

Navigation) and the formal concept lattice is shown and used in the phrase searching. However, all of presented approaches are related to Boolean queries. In terms of the formal framework, Chaudhuri proposed an elegant one to describe query modification, and especially query generalization, for the relational model.

He defined extended queries which express additional constraints on the answer set. Several query modification operators, mainly based on the structure of a query, are defined in order to model constraints which can be added to a query. However, the goal was not to support the refinement of a user's need, but just the extension of the query. Therefore, a generalization contains only one way to modify the query. Beside the difference in defining modification operators, we enable a step-by-step modification in which a user can define on his own which of modification can be relevant for his need] Recently, a framework for the refinement of SQL queries in the multimedia databases was proposed . Query refinement is achieved through relevance feedback where the user judges individual result tuples and the system adapts and restructures the query to better reflect the users information need. In that way a kind of similarity search is achieved. However, the approach does not treat the refinement process formally, but rather as a set of heuristics (like predicate addition or removal) described as query refinement strategies.

Moreover, it does not generate a set of refinements which can support a user in developing ill-defined information needs. Regarding searching in product catalogues the most similar approach is presented in [Ric03]. It is an extension of a mediator architecture that supports the relaxation or tightening of query constraints when no or too many results are retrieved from the catalogue. The query language is a type of Boolean queries suitable for the (web) form based querying against product catalogues. The query tightening is enabled when the cardinality of the resulted set has reached a predefined threshold and it is realized by selecting the most informative, not yet constrained product features. The information content of a feature is defined by measuring its entropy.

APPLICATION

Efficient Search

Refinement strategies also greatly vary between users. Search engines can react differently depending on the user performing the search. A search engine that has a history of a user's queries will be able to offer query assistance suited for that user, or offer helpful suggestions about how the user can improve their searching and refinements. For example, the search engine can suggest stemmed queries to a user who would benefit from stemming refinements, or it might display a message like "We noticed you have been using future tenses in your searches, we suggest changing to present tense for better results."

CONCLUDING REMARKS

In our previous work we have defined an approach for refining (relational) queries which enables a user to navigate through the information content incrementally. In each refinement step a user is provided with a complete but minimal set of refinements, which enables him to develop/express his information need in a step-by step fashion. The approach is based on the model-theoretic interpretation of the refinement problem, such that the query refinement process can be considered as the process of inferring all queries which are subsumed by a given query. In this paper we presented the challenges for a information retrieval process and explained the roles an ontology can play in resolving these problems. In order to illustrate the advantages of the proposed query refinement process using Filters we presented a case

study in which our approach is used as a support for the traditional (free-text based) searching process in a bibliographic information portal. It supports so the called step-by-step query refinement, which enables a novel user to inspect the content of the bibliographic database in a more systematic manner. The evaluation study showed two main advantages of such a refinement: (i) a user can find relevant documents faster and (ii) he is more satisfied with the relevance of the documents for his information need.

REFERENCES

1. P. Anick, Using terminological feedback for web search refinement: a log-based study. In SIGIR '03, 88-95, 2003.
2. R. Baeza-Yates, C. Hurtado, and M. Mendoza, Query recommendation using query logs in search engines. In EDBT' 04, 588-596, 2004.
3. Bruza, P.D. and Dennis, S. Query Reformulation on the Internet: Empirical Data and the Hyper index Search Engine. In RIAO '97, 488-499, 1997.
4. He, D., Göker, A., and Harper, D.J. Combining evidence for automatic web session identification. Information Processing & Management, 38(5), 727-742, 2002.
5. Jansen, B.J., Spink, A., Blakely, C., and Koshman, S. Defining a session on Web search engines. Journal of the American Society for Information Science and Technology, 58(6), 862-871, 2007.
6. Guo, J., Xu, G., Li, H., and Cheng, X. A unified and discriminative model for query refinement. In SIGIR '08,379-386, 2008.
7. Jansen, B.J., Zhang, M., and Spink, A. Patterns and transitions of query reformulation during web searching. International Journal of Web Information Systems, 3(4), 328-340, 2007.
8. Lau, T. and Horvitz, E. Patterns of search: analyzing and modeling Web query refinement. In User Modeling '99,119-128, 1999.
9. Dou, Z., Song, R., Yuan, X., and Wen, J. Are click through data adequate for learning web search rankings?. In CIKM '08, 73-82,2008
10. Efthimiadis, E.N. Query Expansion. Annual Review of Information Science and Technology, 31, 121-187, 1996.
11. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Radlinski, F., and Gay, G. Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search. ACM Transactions on Information Systems, 25(2), 2007.
12. Fox, S., Karnawat, K., Mydland, M., Dumais, S., and White, T. Evaluating implicit measures to improve web search. ACM Transactions on Information Systems, 23(2), 147-168,2005.
13. Rieh, S.Y. and Xie, H. Analysis of multiple query reformulations on the web: the interactive information retrieval context. Information Processing & Management, 42(3), 751-768, 2006.